# Study of Best Model for Diabetes Prediction using Feature and Model Selection

## Introduction

Diabetes can develop when the pancreas does not produce enough insulin or when the insulin produced is not utilized by the body. It is a chronic illness and there is no cure. Early detection and predicting using diagnostic measures is important to manage the disease and prevent or reduce the risk of developing other complications.

## Dataset

## The dataset was downloaded from Kaggle (<https://www.kaggle.com/uciml/pima-indians-diabetes-database>)

* The data has 8 numerical features with Size : 768 rows and 9 columns
* No null values
* Class variable: Outcome (0 – no diabetes, 1- diabetes)
* Class imbalances in the data: Negative labels almost double that of positive

## Project Goal

## The goal of the project is to determine the best model and features to correctly predict if a person has diabetes based on diagnostic data.

## Methodology

The methodology included the following steps as sown in the figure below.

A diagram of a system

Description automatically generated

* I did the data overview, EDA, imputing zero values based on feature histograms, feature selection based on correlation with class label and feature scaling as data range for features were varied (details in power point)
  + Selected Features were Glucose, BMI, Age, Diabetes Pedigree Function
  + Pregnancies was not selected as it had a high correlation with Age
* Before imputing the values the dataset was split into train and test, scaled and train and test data imputed separately. Zero values for:
  + Insulin, and Skin Thickness imputed to median (skewed distribution)
  + 'Glucose', 'Blood Pressure', 'BMI ’imputed to mean (normal distribution)
* I then ran each model individually once for ‘All Features’ and once for ‘Selected Features’
* I tuned the hyperparameters for kNN, Logistic Regression and Random Forest iteratively and used GridSearchCV for SVM
* As there was class imbalance I set the parameter class\_weight = balanced for all models except kNN(does not have this parameter)
* Computed performance metrics for each model for each run with best hyperparameters

## Hyper Parameter Tuning

Performed for all 4 models- once for ‘All Features’ and once for ‘Selected Features’

K-Nearest Neighbors

* Tuned parameter k for number of neighbors for best accuracy and recall

Logistic Regression

* Tuned hyperparameter C to find the best C for highest accuracy
  + C - Controls the strength of the regularization
* Set class\_weights = balanced

Random Forest

* Tuned hyperparameters N = number of trees and d = maxdepth for lowest error rate
* Set class\_weights = balanced to remedy class imbalance

SVM

* Used GridSearchCV to get the best value of C and gamma
  + C - Controls the strength of the regularization
  + Gamma - inverse of the radius of influence
* Set class\_weights = balanced to remedy

## Best Hyperparameters

|  |  |  |
| --- | --- | --- |
| **Model** | **Features** | **Parameters** |
| **kNN** | **ALL** | best k = 19 |
| **kNN** | **Selected** | best k = 17 |
| **Logisitc Regression** | **ALL** | best C: 1 |
| **Logisitc Regression** | **Selected** | best C: 0.001 |
| **SVC(GridSearch)** | **ALL** | best params : C =100,gamma= 0.001,class\_weight = 'balanced',kernel = 'rbf' |
| **SVC(GridSearch)** | **Selected** | best params: C=10, gamma=0.01,class\_weight='balanced',kernel = 'rbf' |
| **Random Forest** | **ALL** | best N: 2 Best d: 6 |
| **Random Forest** | **Selected** | best N: 9 Best d: 3 |

## Results

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Features** | **Accuracy** | **Precision** | **Recall** | **f1\_score** | **TNR** | **TN** | **FP** | **FN** | **TP** |
| **kNN** | **ALL** | 0.7969 | 0.7800 | 0.5821 | 0.6667 | 0.9120 | 114 | 11 | 28 | 39 |
| **kNN** | **Selected** | 0.8177 | 0.7857 | 0.6567 | 0.7154 | 0.9040 | 113 | 12 | 23 | 44 |
| **Logisitc Regression** | **ALL** | 0.7812 | 0.6812 | 0.7015 | 0.6912 | 0.8240 | 103 | 22 | 20 | 47 |
| **Logisitc Regression** | **Selected** | 0.7865 | 0.6857 | 0.7164 | 0.7007 | 0.8240 | 103 | 22 | 19 | 48 |
| **SVC(GridSearch)** | **ALL** | 0.7760 | 0.6765 | 0.6866 | 0.6815 | 0.8240 | 103 | 22 | 21 | 46 |
| **SVC(GridSearch)** | **Selected** | 0.7760 | 0.6667 | 0.7164 | 0.6906 | 0.8080 | 101 | 24 | 19 | 48 |
| **Random Forest** | **ALL** | 0.7969 | 0.7059 | 0.7164 | 0.7111 | 0.8400 | 105 | 20 | 19 | 48 |
| **Random Forest** | **Selected** | 0.7865 | 0.6625 | 0.7910 | 0.7211 | 0.7840 | 98 | 27 | 14 | 53 |

## Analysis

* kNN with selected features has the highest accuracy at 81.77 % and highest precision 78.57% with lower recall 65.7 % and f1\_score of 71.54%. This could be because I could not use a parameter(knn has no class\_weight) to offset the class imbalance.
* Random forest with selected features has the best recall at 79.1 percent and best f1\_score of 72.11%, but lower accuracy 78.65% and lower precision 66.25%.
* Overall Random Forest showed the maximum improvement with ‘Selected Features’ while Logistic Regression showed marginal improvement

## Conclusion

* Best Model: Random Forest Classifier
  + Since my goal is to predict accurately whether a person has diabetes or is likely to get diabetes I would use the Random Forest Model since it has the best recall at 79.1 % and best f1\_score at 72.11%
* Using an ensemble model, the random forest helped to get best recall and f1 score over the individual models.

## Next Steps

Though a recall of 79.1% and f1\_score of 72.11% is not bad I feel we could improve on the performance metrics by:

* Trying out other models (specially ensemble)
* Trying to impute data in with other imputation methods like kNN impute.
* Trying to reduce class imbalance by resampling.

.